

AX.NET Feature Highlights

```
reference to AX.NET...
using ProISV.AX;

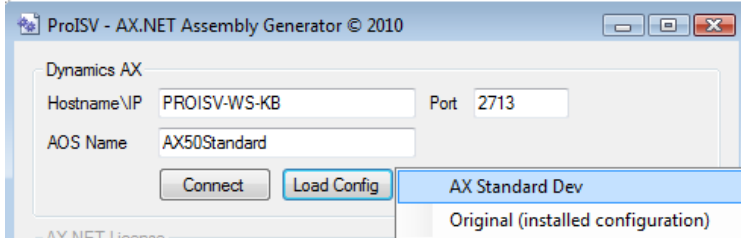
namespace AX.LINQ.Sample1
{
    class Program
    {
        static void Main(string[] args)
        {
            //Create and open connection
            using (AXConnection conn = new AXConnection(new AXEndpoint
                ("AX510Standard", "proisv-ws-kb", 2113, "Demo"), true))
            //Create instance of generated AOT from the AX.NET Ass
            using (AsmGenTest.MyAOT act =
                AsmGenTest.MyAOT.CreateAOT<AsmGenTest.MyAOT>(conn))
            {
                //...
            }
        }
    }
}
```

AX.NET

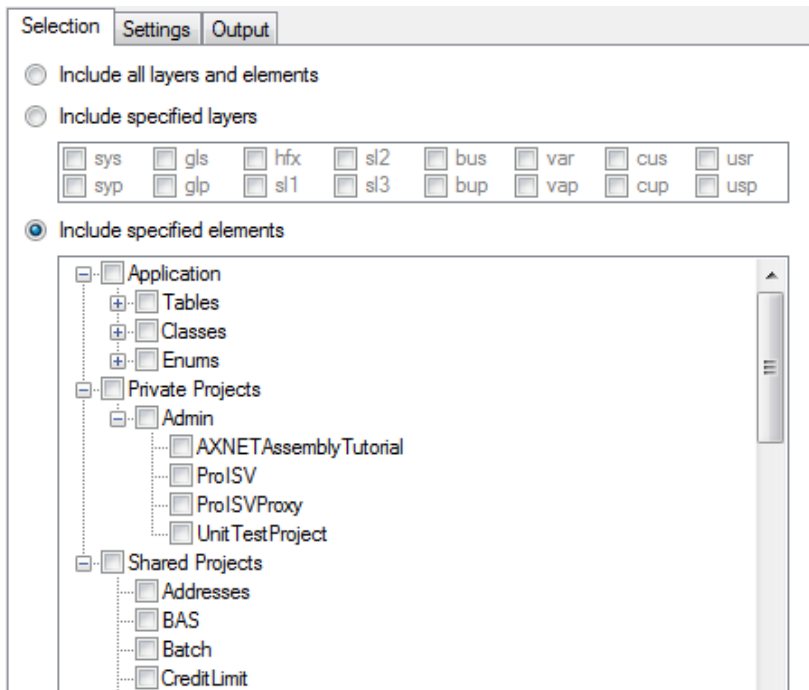
AX.NET Assembly Generator

The AX.NET Assembly Generator Framework is based upon the AX.NET Assembly that is automatically generated by using the AX.NET Assembly Generator.

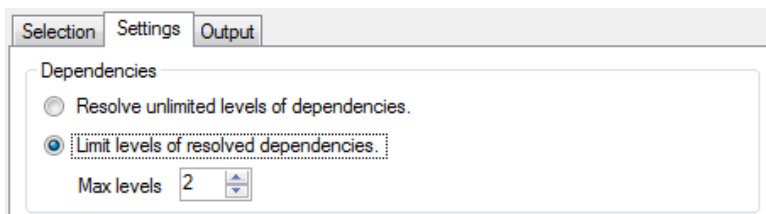
First, the Microsoft Dynamics AX configuration is chosen:



Then the specific Microsoft Dynamics AXA layers and elements are selected to be included in the AX.NET Assembly. This can also include specific AOT projects (private or shared):



Hereafter the level of dependencies is defined, and the AX.NET Assembly Generator will now produce the .NET representation of Microsoft Dynamics AX.



That is how easy it is to move Microsoft Dynamics AX to .NET!

Access AX Business Logic

Upon Assembly Generator completion, all chosen AX logic can now be directly accessed and used for development in .NET, such as the following code example.

```
//Find sale
SalesTableRow sale = (from c in aot.Table.CustTable
    join s in aot.Table.SalesTable
        on c.AccountNum equals s.CustAccount
    orderby s.createdDateTime descending
    where c.Name.Contains("forest")
    select s).First();

//Get exchange rate
double exchangeRate = sale.exchRate();

//Check if above credit limit
bool belowCreditLimit = |
    aot.Class.SalesTableType.checkAgainstCreditLimit(sale);
```

Using AX.NET it's thereby possible to utilize any X++ Business Logic in Dynamics AX from .NET with ease. It is even possible to select data using LINQ and pass the data to a method on a table or class as a parameter. Likewise it's possible to use instances of classes as parameters, when calling methods.

In short AX.NET provides a rich and full integration to all Business Logic in Dynamics AX using simple .NET code.

Access data with LINQ

AX.NET provides support for the new technology in .NET 3.5 called LINQ (Language Integrated Query). LINQ enables easy and structured data access by enabling inline data query to be written with C# code against a model of the data source.

The syntax for writing LINQ queries has many similarities with SQL statements, and is therefore easy for new developers to adapt.

```
//Create LINQ query to search for customers matching hte search string.
var qCustomers = from c in aot.Table.CustTable
    orderby c.AccountNum ascending
    where
        c.AccountNum == searchFor ||
        c.Name.Contains(searchFor)
    select c;
```

Using the AX.NET Assembly Generator that comes with AX.NET, a .NET assembly (dll) containing the structure (tables, classes, enum) of any AX Application can automatically be generated. AX.NET then uses the generated AOT in the AX.NET Assembly to provide the data model for the LINQ statements.

It is therefore possible to use .NET LINQ to query any data in an AX Application including custom tables and fields, because the AX.NET Assembly is tailored to match any AX Application.

Product Sheet

The support for LINQ in AX.NET is a native implementation, because it transforms the LINQ statements to X++ select statements and executes them through the Microsoft Dynamics AX Business Connector. There is no direct communication with the Microsoft Dynamics AX database, so all Dynamics AX Business Logic and security is enforced.

Furthermore AX.NET also supports insert, update and delete of data in Microsoft Dynamics AX. All data retrieved using LINQ can easily be modified and deleted in .NET and the changes can be saved in Microsoft Dynamics AX.

The following LINQ features are supported:

Select, Join, Where, Order By, Group By, First, FirstOrDefault, Max, Min, Average, Sum, Count, SelectForUpdate.

Efficient Development in Visual Studio

When integrating to Microsoft Dynamics AX you have to use the Microsoft Dynamics AX Business Connector. The Business Connector does provide access to the Microsoft Dynamics AX Application, but the surface / interface of the Business Connector does not reflect the interface of the Microsoft Dynamics AX Application.

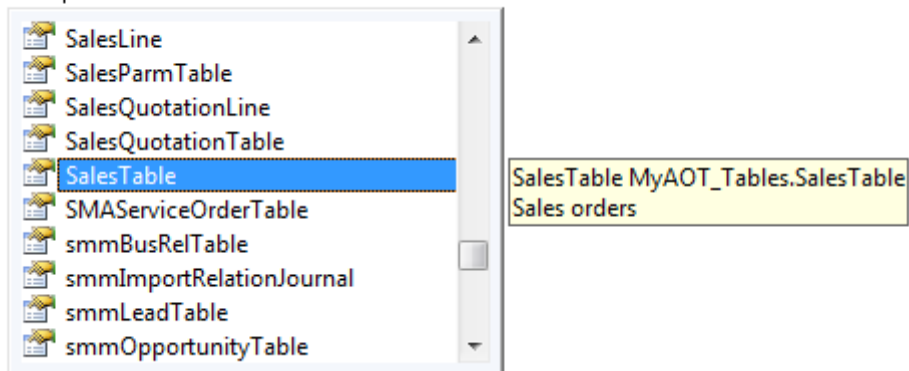
This results in the lack of support for the modern productivity enhancing features of Visual Studio like Intellisense, automatic syntax and compilation verification.

If a change is made to the Microsoft Dynamics AX Application that changes the interface (like renaming a class or table field), there will be no compilation errors generated in the Visual Studio Solution, because the Business Connector does not provide the compiler any information about what is correct and valid in regards to the Microsoft Dynamics AX Application.

With AX.NET this all changes! The AX.NET technology is based upon the AX.NET Assembly that is automatically generated by the AX.NET Assembly Generator. The AX.NET Assembly contains a .NET version of the AOT representing a Microsoft Dynamics AX Application.

When you use AX.NET to integrate with Microsoft Dynamics AX, you are actually working on real .NET objects and enums, and thereby releasing all the productivity enhancing features of Visual Studio, e.g. browsing through the AOT,

aot.Table. |

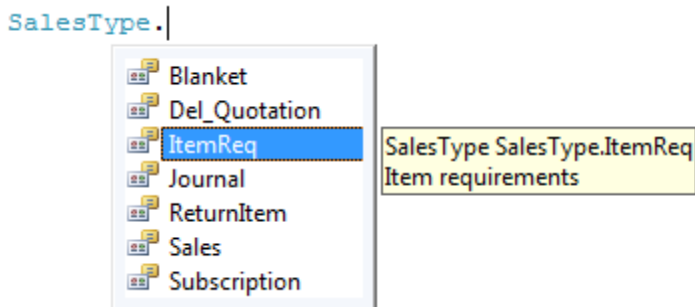


or providing Intellisense and auto completing, when writing code in Visual Studio.

```
aot.Table.CustTable.findByPartyId(|  
▲ 2 of 2 ▼ CustTableRow CustTable.findByPartyId (string _partyId, bool _forUpdate)
```

Product Sheet

All labels from Microsoft Dynamics AX are automatically placed on Enums and Tables and are available as documentation in Visual Studio.



AX.NET provides full support for labels in Microsoft Dynamics AX, and thereby making it much easier to maintain localization.

```
//Retrieve english label for ReturnItem on the SalesType enum.  
string enumLabel = aot.LabelManager.GetLabel  
(  
    SalesType.ReturnItem, "en-us"  
);
```

Using AX.NET it's possible to avoid having labels stored in both Microsoft Dynamics AX and .NET, because all localization and labels can easily be handled by Microsoft Dynamics AX.

Label Support in AX.NET

AX.NET also fully supports Microsoft Dynamics AX labels, including all label files as well as labels on enums and table fields. The label ID's are stored in the AX.NET Assembly generated by the AX.NET Assembly Generator.

The AX.NET Label Manager makes it easy to retrieve label texts. In this example a label text of an enum value is retrieved:

```
//Get label of dimension using default language.  
string lbl = AOT.LabelManager.GetLabel(AXStd.Enums.SysDimension.Center);
```

Labels can be retrieved in several other ways, such as labels on a table field, labels in a specific language, or manual label retrieval by specifying the label ID.

About ProISV

ProISV is a company headquartered in Copenhagen, Denmark and is founded by executives having a long track record in developing software in all categories, including .NET, Microsoft Dynamics AX and other Microsoft as well as open source technologies.

ProISV believes the future belongs to Cloud computing and development standardization in .NET. Therefore, the first suite of products from ProISV is designed to move Microsoft Dynamics AX into .NET and to be fully Cloud enabled.

Contact information:

ProISV ApS
C.F. Møllers Allé 20, 6. Th.
2300 Copenhagen S
Denmark

Sales & distribution inquiries: ns@proisv.com

Contact phone number: +45 20 26 93 94